Computing Join Aggregates over Private Tables

Rong She[1], Ke Wang[1], Ada Waichee Fu[2], Yabo Xu[1]

[1] School of Computing Science, Simon Fraser University, Canada
{rshe,wangk,yxu}@cs.sfu.ca
[2] Dept. of Computer Science & Engineering, Chinese University of Hong Kong
adafu@cse.cuhk.edu.hk

Abstract. We propose a privacy-preserving protocol for computing aggregation queries over the join of private tables. In this problem, several parties wish to share aggregated information over the join of their tables, but want to conceal the details that generate such information. The join operation presents a challenge to privacy preservation because it requires matching individual records from private tables. We solve this problem by a novel sketching protocol that securely computes some randomized summary information over private tables. It ensures that during the query computation process, no party will learn other parties' private data, including the individual records and data distributions. Previous works have not provided this level of privacy for such queries.

1 Introduction

In many scientific and business applications, collaborations among various autonomous data holders are necessary in order to obtain global statistics or discover trends. When private sources are involved in such practices, their privacy concerns must be addressed. For example, a hospital keeps a table H of patients' medical histories, and a research institute has a table R of patients' DNA samples. Both tables contain a common attribute "Patient_Name" (not necessarily a key attribute if a patient has several diseases or DNA samples). To establish the relationships between diseases and DNA anomalies, the research institute wants the answer to the following query:

```
SELECT      H.Disease, R.DNA_Characteristics, COUNT(*)
FROM        H, R
WHERE       H.Patient_Name = R.Patient_Name
GROUP BY    H.Disease, R.DNA_Characteristics
```

This query returns the number of occurrences for each combination of disease and DNA characteristics, providing helpful insights into their relationships. For example, from <*d*,*c*,70> and <*c*,100>, where *d* is a disease and *c* is a DNA characteristic, it can be learnt that if a patient has the DNA characteristic *c*, he/she has the disease *d* with a 70% chance. While there is need to answer such queries, due to privacy restrictions such as the HIPAA policies (http://www.hhs.gov/ocr/hipaa/), neither party is willing to disclose local patient-specific information (such as patient names) to the other party. This is an example of *private join aggregate queries* that we want to consider in this paper.

Private Join Aggregate Queries. In general, a *join aggregate query* has the form:

SELECT	group-by-list, *agg*(*exp*)
FROM	$T_1,\ldots,T_n$
WHERE	$J_1=J_1'$ and … and $J_m=J_m'$
GROUP BY	group-by-list

where each T_i is a table; J_i and J_i' are join attributes from different tables; group-by-list is a list of group-by attributes possibly from different tables; *exp* is an arithmetic expression over aggregation attributes; *agg* is an aggregation function. In this paper, we consider the aggregation functions COUNT and SUM. *Conceptually*, the tables in FROM are joined according to the predicates in WHERE, and the joined records are then grouped on the group-by-list. The query result contains one row for each group with *agg*(*exp*) being computed over the group. If the optional GROUP BY is missing, there is only one group and one row in the result. We can assume that a table contains only join attributes, aggregation attribute and group-by attributes; all other attributes are invisible to the query and thus are safe to be ignored for our purpose.

In a *private join aggregate query*, each T_i is a private table owned by a different party. These parties wish to compute and share the query result as specified in SELECT, but not any other information, including join attribute values and their distributions. This level of privacy was not provided by previous solutions, which will be discussed in more details in Section 2.

We assume the honest-but-curious behavior [12]: The parties follow the protocol properly with the exception that they may keep track of all intermediate computations and received messages, and try to induce additional information. Our focus is on privacy protection in the computation process, not against the answers to queries. Our assumption is that, when the participating parties wish to share the query result, information inferred from such results is a fair game [2]. Protection against query results has been studied in statistical databases [1] and is beyond the scope of this paper.

Our Contributions. We present a novel solution to private join aggregate queries based on the sketching technique previously studied for join size estimation and data stream aggregations [4][7]. None of these works involves privacy issues. The basic idea of sketching is that each table maintains a summary structure, called *atomic sketch*, which is later combined to estimate the query result. A striking property of atomic sketches is that they are computed locally without knowing any data in other tables, which makes them promising for privacy protection. However, we will show that a straightforward way of combining atomic sketches would allow a party to learn the distribution of join values owned by other parties. We will analyze the source of such privacy leakage and determine the types of information that need to be concealed. We then propose a private sketching protocol where each party holds a "random share" of the same atomic sketch so that collectively they represent the atomic sketch, but individually they are useless. We show how the query can be estimated directly from such random shares in a way such that no party learns private information (both individual values and their distributions) from other parties.

Due to space limit, we will mainly discuss our protocol for computing the join ag-

gregate COUNT(*) with two parties. It should be noted that this protocol is extendable to multiple parties and on more general aggregates.

2 Related Work

In general secure computations, the trusted third party model [14] allows all parties to send data to a "trusted" third party who does the computation. Such a third party has to be completely trusted and is difficult or impossible to find. In the secure multi-party model [21], given two parties with inputs x and y, the goal is to compute a function $f(x,y)$ such that the two parties learn only $f(x,y)$ and nothing else. In theory, any multi-party computation can be solved by simulating a combinatorial circuit. However, its communication cost is impractical for data intensive problems.

In [8], privacy-preserving cooperative statistical analysis was studied for vertically or horizontally partitioned data. With vertically partitioned data, n records $\{(x_1,y_1),\ldots, (x_n,y_n)\}$ are distributed at two parties such that Alice holds $\{x_1,\ldots, x_n\}$ and Bob holds $\{y_1,\ldots,y_n\}$. In this case, the join relationship is one-to-one and is implicit by the sequential ordering of records. A similar data partition based on a common key identifier is assumed in [9][19]. In real world, it is odd that the data owned by two mutually un-trusted parties are about the exactly same set of entities. In [3], horizontally partitioned data is considered where each party possesses some records from the same underlying table. In this paper, we consider general join relationships specified by the SQL statement, which can be foreign-key based join or many-to-many join.

Another recent work [10] also discussed aggregations such as SUM queries over several private databases. However, it assumes all parties contain the same pair of attributes: a "key" and a "value" field. The SUM query is to aggregate the values with the same key from all parties. In contrast, we deal with tables with different attributes and general join relationships, where the aggregation is defined over the joined table. The closest work to ours is [2] that studied the private join size problem. It proposed a scheme for encrypting join values but required exchanging the frequency of encrypted values. As noted in [2], if the frequency of some join values is unique, the mapping of the encryption can be discovered by matching the frequency before and after the encryption. So the privacy of join values is compromised. We do not have such problem.

Another related problem is the restriction-based inference control in OLAP queries [20][22]. The goal of inference control is to prevent values of sensitive data from being inferred through answers to OLAP queries. These works mainly dealt with the privacy breaches that arise from the answers to multiple queries; they do not consider the privacy breaches during query processing. The inference control problem has been studied largely in statistical databases, for example, see [1].

3 Preliminaries

First, we review some basic techniques that are the building blocks of our solution.

Sketching. Sketching is a randomized algorithm that estimates the join aggregate

result with a random variable, called *sketch*. The sketch is obtained by multiplying some *atomic sketches*, which are computed at each table. The expected value of the sketch is shown to be equal to the aggregate result with bounded variance [4][7].

As an example, consider the query SUM(A) over 3 tables T_1, T_2 and T_3, with join conditions $T_1.J_1=T_2.J_2$ and $T_2.J_3=T_3.J_4$, where A is an aggregation attribute in T_1. The table containing A (in this case, T_1) will be called the *aggregation table*. Each pair of the join attributes (J_1, J_2) or (J_3, J_4) is called a *join pair*. J_2 and J_3 may be the same attribute in T_2, but conceptually they belong to different join pairs and are treated separately. Let D_i denote the domain of J_i. Each join pair shares the same domain. For simplicity, we assume $D_i=\{1,\ldots,|D_i|\}$. For any table T_i, let JS_i be the set of join attributes in T_i. Suppose JS_i contains *m* join attributes, then a *value instance* V on JS_i is a set that contains one value for each of the *m* attributes, i.e. $V=\{x_1,\ldots,x_m\}$ where x_i is a value of a distinct join attribute in JS_i. Let $T_i(V)$ be the set of records in T_i having the value instance V on JS_i. For an aggregation table T_i, we define $S_i(V)$ to be the sum of aggregation attribute values over all records in $T_i(V)$; for any non-aggregation table T_i, we define $F_i(V)$ to be the number of records in $T_i(V)$. Thus, T_1 will have $S_1(V)$ defined; T_2 and T_3 will have $F_2(V)$ and $F_3(V)$ defined. The sketch is constructed as follows:

The ε family: For each join pair (J_i, J_j), select a family of 4-wise independent binary random variables $\{\varepsilon_k, k=1,\ldots,|D_i|\}$, with each $\varepsilon_k \in \{1,-1\}$. That is, each join value k is associated with a variable ε_k whose value is randomly selected from {1,-1} and any 4 tuple of such ε variables is jointly independent. The set of values for all ε_k variables is called a ε family. In this example, there are two independent ε families, one for each join pair. In table T_i, with a join value instance V, for each join value x in V (x is a value of some join attribute J), there is one ε_x variable from J's ε family. Let $E_i(V)= \prod_{x \in V} \varepsilon_x$.

Atomic sketches: There is one atomic sketch for each table. For the aggregation table T_1, its atomic sketch $X_1=\Sigma_V[S_1(V)\times E_1(V)]$, i.e. the sum of $S_1(V)\times E_1(V)$ over all distinct V in T_1, called *S-atomic sketch* (S for summary); the atomic sketch for T_2 and T_3 is $X_2=\Sigma_V[F_2(V)\times E_2(V)]$ and $X_3=\Sigma_V[F_3(V)\times E_3(V)]$, called *F-atomic sketches* (F for frequency).

The sketch: The sketch is defined as $\prod_i (X_i)$, the multiplication of atomic sketches over all tables. The expected value of the sketch can be shown to be equal to SUM(A) with bounded variance. We refer interested readers to [4][7] for details.

Because sketch is a random variable, the above computation must be repeated many times to get a good average. [5] suggests a procedure of boosting where the number of trials is α×β. For every α trials, the average of their sketches is computed, resulting in β averages. The final estimator is the median of these β averages. Note the ε families are chosen independently in each trial. We will refer to this process as αβ-boosting. The time complexity of sketching with αβ-boosting is $O(\alpha\times\beta\times\Sigma_i|T_i|)$, where $|T_i|$ denotes the number of records in table T_i. Experiments from previous works and our experiences show it is usually accurate (error rate < 5%) with moderate size of α (~50) and β (~5).

Private Shared Scalar Product Protocol. The *private scalar product* protocol was first discussed in [8]. Given two d-dimensional vectors $\vec{U}=<U_1,\ldots,U_d>$ and $\vec{V}=<V_1,\ldots,V_d>$ owned by two honest-but-curious parties, this protocol computes $\vec{U}\times\vec{V}$ such that the two parties obtain no additional knowledge other than $\vec{U}\times\vec{V}$.

In some applications, a scalar product $\vec{U}\times\vec{V}$ is needed as a part of the computation,

but the value of $\vec{U} \times \vec{V}$ needs to be concealed. Such problems can be addressed by the *private shared scalar product* (SSP) protocol [11]. Each party obtains a random share of $\vec{U} \times \vec{V}$, denoted as R_1 and R_2, such that $R_1+R_2=\vec{U} \times \vec{V}$. R_1 and R_2 are complementary to each other with their sum being $\vec{U} \times \vec{V}$, but the sum is unknown to both parties. The range of R_1 and R_2 can be the real domain, thus it is impossible to guess $\vec{U} \times \vec{V}$ from any single share. Efficient two-party SSP protocols are available with linear complexity [6][8][9]. The multi-party SSP protocol was studied in [11]. In the rest of this paper, we will use SSP($\vec{V}_1,\ldots,\vec{V}_k$) to denote the SSP protocol on input vectors $\vec{V}_1,\ldots,\vec{V}_k$.

4 Private Sketching Protocol

We illustrate our protocol on the basic join aggregate COUNT(*) over two private tables, i.e. the join size of two tables. Note that our protocol can be extended to other queries. We will first analyze the privacy breaches in the standard sketching process, from which we derive the requirements on the types of information that must be concealed. We then show how to conceal such information using our protocol.

Assume that Alice holds table T_1 and Bob holds T_2 with a common join attribute J. Let D_1 be the set of join values in T_1, D_2 be the set of join values in T_2. Thus, J's active domain $D= D_1 \cup D_2$. First, a ε family for J is selected. Then both parties use this same ε family to compute their atomic sketches. For illustration purposes, assume D has two values v_1 and v_2 ($|D|=2$); there are two variables $\{\varepsilon_1, \varepsilon_2\}$ in the ε family. Let $F_i(v)$ denote the number of records with join value v in table T_i. $F_1(v)$ belongs to Alice and should be concealed from Bob; $F_2(v)$ belongs to Bob and should be concealed from Alice.

The two parties compute their atomic sketches X_i as follows:

Alice (T_1): $$X_1 = F_1(v_1) \times \varepsilon_1 + F_1(v_2) \times \varepsilon_2, \quad (1)$$

Bob (T_2): $$X_2 = F_2(v_1) \times \varepsilon_1 + F_2(v_2) \times \varepsilon_2. \quad (2)$$

So far, the computation of X_i is done locally, using the shared ε family and locally owned $F_i(v_j)$ values without any privacy problem. Because the ε family is just some random value, knowing it will not lead to any private information about the other party.

Next, the sketch $X_1 \times X_2$ needs to be computed. Suppose Alice sends her atomic sketch X_1 to Bob. Now, Bob knows X_1, X_2 and the ε family. In Equation (1) and (2), with only $F_1(v_1)$ and $F_1(v_2)$ being unknown, Bob can infer some knowledge about $F_1(v_j)$. For example, knowing $\varepsilon_1=1$ and $\varepsilon_2=-1$, if X_1 is positive, Bob knows that v_1 is more frequent than v_2 by a margin of X_1 in T_1. The problem may be less obvious when there are more values in D, however, it still leaks some hints on $F_1(v_j)$. Furthermore, in the αβ-boosting process, the above computation is repeated α×β times and there is one pair of Equation (1) and (2) for *each* of the α×β trials. With the ε family being independently chosen in each trial, each pair of equations provides a new constraint on the unknown $F_1(v_j)$ values. If the number of trials is equal to or greater than |D|, Bob will have a sufficient number of Equation (1) to solve all $F_1(v_j)$ values. Therefore, even for a large domain D, the privacy breach is severe if Bob knows both atomic sketches.

On the other hand, even if Bob only knows his own X_2, given the result of $X_1 \times X_2$, he can easily get X_1. Now, if the individual sketch $X_1 \times X_2$ in each trial is also concealed

from Bob, because both parties agree to share the final result which is an average of $X_1 \times X_2$, by comparing the final result with his own X_2's, Bob may still infer some approximate knowledge on X_1. The situation is symmetric with Alice. Therefore, to prevent any inference on other party's $F_i(v_j)$, all atomic sketches should be unknown to all parties. This implies that the ε families should also be concealed from all parties.

Now suppose all atomic sketches X_i and ε families are concealed. If the sketch $X_1 \times X_2$ is known to Bob, Bob may still learn X_1 in some extreme cases. For example, knowing $X_1 \times X_2 = 0$, and $F_2(v_1) = 10$ and $F_2(v_2) = 5$, since $X_2 \neq 0$ for any value of ε_1 and ε_2, Bob can infer that $X_1 = 0$. Additionally, from Equation (1), $X_1 = 0$ holds only if $F_1(v_1) = F_1(v_2)$ (because $\varepsilon_1, \varepsilon_2 \in \{1, -1\}$). Consequently, Bob learns that the two join values are equally frequent in T_1. To prevent this, the individual sketch in each trial should also be concealed from all parties. Therefore, the only non-local information that a party is allowed to know is the final query result which will be shared at the end. Because the final result is something that has been averaged over many independent trials, disclosing one average will not let any party infer the individual sketches or underlying atomic sketches. Note that with the current problem definition where parties agree to share the final result, we cannot do better than this.

Since the ε families must be concealed from Alice and Bob, we need a semi-trusted third party [15], called Tim, to generate the ε families. To fulfill its job, Tim must also be an honest-but-curious party who does not collude with Alice or Bob. In real world, finding such a third party is much easier than finding a trusted third party. The protocol must ensure that Tim does not learn private information about Alice or Bob or the final query result, i.e. Tim knows nothing about atomic sketches, sketches or $F_i(v_j)$ values. The only thing Tim knows is the ε families which are just some random variables. On the other hand, Alice owns $F_1(v_j)$ and Bob owns $F_2(v_j)$, both should know nothing about the other party's $F_i(v_j)$, the ε families, atomic sketches or individual sketches.

Information Concealing. Let Y denote an average of sketches over α trials in αβ-boosting. There will be β number of such Y's in total. A protocol satisfying the following requirements is called *IC-conforming*: Alice learns only Y's and local $F_1(v_i)$; Bob learns only Y's and local $F_2(v_i)$; Tim learns only the ε families; atomic sketches X_i and individual sketches $X_1 \times X_2$ are concealed from all parties.

Theorem 1. A IC-conforming protocol conceals $F_i(v)$ from all non-owning parties throughout the computation process. ■

Proof: First, Tim knows only the ε families and nothing about $F_i(v)$. Consider Alice and Bob. From IC-conformity, the only non-local knowledge gained by Alice or Bob is the value of Y, which is an average of sketches over α trials. With $\alpha \geq 2$, such an average provides no clue on any individual sketch because each sketch is computed with an independent and random ε family. Even if there is a non-zero chance that Tim chooses the same ε family in all α trials, therefore Y is equal to each individual sketch, Alice or Bob will have no way of knowing it because the ε families are unknown to them.

Alice or Bob knows that Y is an approximation of the query result $F_1(v_1) \times F_2(v_1) + \ldots + F_1(v_k) \times F_2(v_k)$. However, this approximation alone does not allow any party to solve the other party's $F_i(v)$ because there are many solutions for the unknown $F_i(v)$. It does

not help to use different averages Y in αβ-boosting because they are instances of a random variable and do not act as independent constraints. ■

4.1 IC-Conforming Protocol for Two-Party COUNT(*) Query

Assume Bob is the querying party who issues the query. The overall process of our protocol is shown below. The α×β trials are divided into β groups, each containing α trials. Each trial has the ε-phase and the S-phase. The ε-phase generates the ε family and the S-phase computes atomic sketches. For each group, the α-phase computes the sketch average over α trials. Finally the β-phase finds the median of the β averages.

```
1. for i=1 to β do
2.      for j=1 to α do
3.          ε-phase;
4.          S-phase;
5.      α-phase;
6. β-phase;
```

ε-phase. In this phase Tim generates the ε family. Let D_1 be the set of join values in T_1 (Alice's table); D_2 be the set of join values in T_2 (Bob's table). $D=D_1 \cup D_2$. To generate the ε family, Tim needs $|D|$, $|D_1|$, $|D_2|$ and the correspondence between ε variables and join values. Alice and Bob can hash their join values by a cryptographic hash function H [18]. H is (1) *pre-image resistant*: given a hash value H(v), it is computationally infeasible to find v; (2) *collision resistant*: it is computationally infeasible to find two different inputs v_1 and v_2 with $H(v_1)=H(v_2)$. Industrial-strength cryptographic hash functions with these properties are available [16]. The ε-phase is as follows.

1. Alice and Bob agree on some cryptographic hash function H and locally compute the hashed sets of their join values $S_1=\{H(v)|v \in D_1\}$ and $S_2=\{H(v)|v \in D_2\}$ using H.

2. Alice sends S_1 and Bob sends S_2 to Tim.

3. Tim computes $S=S_1 \cup S_2$.

4. Tim assigns a unique ε variable to each value in S, generating a ε family $\vec{E}_1$ for S_1 and a separate ε family $\vec{E}_2$ for S_2.

Security analysis. Alice and Bob do not receive information from any party. With the cryptographic hash function H, Tim is not able to learn original join values from the hashed sets. Since Tim does not know H, it is impossible for Tim to infer whether a join value exists in T_1 or T_2 by enumerating all possible values. What Tim does learn is the domain size $|D_1|$, $|D_2|$, $|D_1 \cup D_2|$ and $|D_1 \cap D_2|$. But they will not help Tim to infer atomic sketches or sketches. Therefore, this phase is IC-conforming.

S-phase. This phase computes the atomic sketches X_1 for T_1 and X_2 for T_2. Let $\vec{F}_i$ be the vector of $F_i(v)$ values where $v \in D_i$, arranged in the same order as in $\vec{E}_i$. X_i is actually the scalar product $\vec{F}_i \times \vec{E}_i$ where $\vec{F}_i$ is owned by T_i and $\vec{E}_i$ is owned by Tim. To conceal $\vec{E}_i$, $\vec{F}_i$ and X_i, the three parties can use SSP protocol to compute X_i.

1.Alice and Tim compute SSP($\vec{E}_1, \vec{F}_1$), where Alice obtains RA and Tim obtains TA, with $RA+TA=X_1$.

2.Bob and Tim compute SSP($\vec{E}_2, \vec{F}_2$), where Bob obtains RB and Tim obtains TB,

with RB+TB=X_2.

Security analysis. The SSP protocol ensures that $\vec{E}_i$, $\vec{F}_i$ and X_i are concealed. Tim obtains two non-complementary random shares of different atomic sketches, which are not useful to infer any atomic sketch. Thus, this phase is IC-conforming.

α-phase. This phase computes the average of sketches for every α trials. The sketch in the jth trial is $X_{1j} \times X_{2j}$, where X_{1j} and X_{2j} are atomic sketches for T_1 and T_2. However, at the end of S-phase, no party knows X_{1j} or X_{2j}; rather, Tim has TA_j and TB_j, Alice has RA_j and Bob has RB_j, such that $X_{1j}=TA_j+RA_j$ and $X_{2j}=TB_j+RB_j$. After α trials, let $\overrightarrow{RA}$ be the vector $<RA_1,\ldots,RA_\alpha>$ and let $\overrightarrow{RB}, \overrightarrow{TA}, \overrightarrow{TB}$ be defined analogously. Alice owns $\overrightarrow{RA}$, Bob owns $\overrightarrow{RB}$, Tim owns $\overrightarrow{TA}$ and $\overrightarrow{TB}$. The sketch average Y over the α trials is:

$$Y = \frac{\sum_{j=1}^{\alpha}(X_{1j} \times X_{2j})}{\alpha} = \frac{\sum_{j=1}^{\alpha}[(RA_j + TA_j) \times (RB_j + TB_j)]}{\alpha}$$

$$= \frac{\sum_{j=1}^{\alpha}(RA_j \times RB_j + TA_j \times RB_j + RA_j \times TB_j + TA_j \times TB_j)}{\alpha} = \frac{\overrightarrow{RA} \times \overrightarrow{RB} + \overrightarrow{TA} \times \overrightarrow{RB} + \overrightarrow{RA} \times \overrightarrow{TB} + \overrightarrow{TA} \times \overrightarrow{TB}}{\alpha}$$

The numerator is the sum of several scalar products. To compute these scalar products, if we allow the input vectors to be exchanged among parties, a party obtaining both complementary random shares immediately learns the atomic sketch, thereby violating the IC-conformity. Therefore we use the SSP protocol again as follows.

1. Alice and Bob compute SSP($\overrightarrow{RA}, \overrightarrow{RB}$).
2. Tim and Bob compute SSP($\overrightarrow{TA}, \overrightarrow{RB}$).
3. Tim and Alice compute SSP($\overrightarrow{TB}, \overrightarrow{RA}$).
4. Tim computes $\overrightarrow{TA} \times \overrightarrow{TB}$ (no SSP is needed).
5. Tim sums up all his random shares and $\overrightarrow{TA} \times \overrightarrow{TB}$, sends the sum to Alice.
6. Alice adds all her random shares to the sum from Tim, forwards it to Bob.
7. Bob adds all his random shares to the sum from Alice, divides it by α. In the end, Bob has the average Y over the α trials.

Security analysis. The SSP protocols ensure that $\overrightarrow{RA}, \overrightarrow{RB}, \overrightarrow{TA}, \overrightarrow{TB}$ are concealed from a non-owning party; therefore, no party learns atomic sketches. After SSP computations, a party may obtain several non-complementary random shares. For example, Alice obtains one random share of $\overrightarrow{RA} \times \overrightarrow{RB}$ and one random share of $\overrightarrow{TB} \times \overrightarrow{RA}$, which will not help her learn anything. A party may receive a partial sum during sum forwarding. However, each partial sum always contains two or more non-complementary random shares. It is impossible for the receiver to deduce individual contributing random shares from such a sum. Therefore, this phase is IC-conforming.

β-phase. Repeating the α-phase β times would yield the averages $Y_1, \ldots, Y_\beta$ at Bob. In the β-phase, Bob finds the median of them, which is the final query estimator.

Security analysis. This phase is done entirely by Bob alone and there is no information exchange at all. Thus the level of privacy at all parties is unchanged.

Cost Analysis. Let $|T_i|$ be the number of records in T_i. Let C_H denote the computation cost of one hash operation. Let $C_{SP}(d)$ denote the computation cost and S(d) denote the communication cost for executing the SSP protocol on d-dimensional vectors.

The running time of our protocol is as follows. (1) *ε-phase*: hashing and generating ε families takes $O(C_H \times |D| + \Sigma_i |T_i| + \alpha \times \beta \times |D|)$ time. Note that hashing is done only once for all trials, but the ε family is generated independently in each trial. (2) *S-phase*: computing atomic sketches takes $O(\alpha \times \beta \times \Sigma_i C_{SP}(|D|))$. (3) *α-phase*: computing the β averages takes $O(\beta \times C_{SP}(\alpha))$. (4) *β-phase*: finding the median of β averages takes $O(\beta)$ time.

The communication cost is $2|D|$ for generating ε families, $2\alpha\beta \times S(|D|)$ for computing atomic sketches, $3\beta \times S(\alpha)$ for computing averages.

4.2 Protocol Extensions

Our protocol can be extended to n-party queries. Using only one third party Tim, each party securely computes their atomic sketches with Tim and the sketch averages are computed over n parties using the n-party SSP protocol. Our protocol also works with SUM(A) queries. The only difference is that the table with attribute A will compute S-atomic sketches instead of F-atomic sketches. Such change only affects local computations in that table. We can even handle more general forms of aggregations like SUM(A×B×C) or SUM(A+B+C). Group-by operators can also be handled, where each group can be considered a partition of original tables and there is a sketch for each partition. In addition, our protocol is extendable to perform roll-up/drill-down operations [13], by rolling-up/drilling-down on local random shares. For details on our protocol extensions, please refer to a full version of our paper [17].

5 Experiments

We implemented the two-party protocol on three PCs in a LAN to simulate Alice (T_1), Bob (T_2) and Tim. All PCs have Pentium IV 2.4GHz CPU, 512M RAM and Windows XP. The cryptographic hash function was implemented using QuickHash library 3.0 (http://www.slavasoft.com/quickhash/). We use the SSP protocol in [9]. Tests were done on synthetic datasets with various table sizes and join characteristics. $|D|$ varies from 100 to 10000, $|T_1|$ from 10000 to 1 million, join values follow zipf distribution. T_2 was generated such that for every join value in T_1, B number (1~10) of records are generated in T_2 with the same join value. Thus, $|T_2| = B \times |D|$ and the join size $|T_1 \infty T_2| = B \times |T_1|$. In our experiments, with α ranging from 50 to 300 and β from 5 to 20, the error rate in all runs is no more than 6%. For large α and β, the error is usually less than 2%. This shows that the approximation provided by sketching is sufficient for most data mining applications where the focus is on trends and patterns, instead of exact counts. As analyzed in Section 4, the protocol is very efficient and finishes within seconds in all runs. Please see [17] for more details.

6 Conclusions

We proposed a privacy-preserving protocol for computing join aggregate queries over private tables. The capabilities of computing such queries are essential for collaborative data analysis that involves multiple private sources. By a novel transformation of

the sketching technique, we achieve a level of protection not provided by the previous encryption method. The key idea is locally maintaining random shares of atomic sketches that provide no clue on the data owned by other parties.

References

1. N. R. Adam, J. C. Wortman, Security-control methods for statistical databases, ACM Computing Surveys, 21(4):515-556 (1989).
2. R. Agrawal, A. Evfimievski, R. Srikant, Information sharing across private databases, SIGMOD (2003)
3. R. Agrawal, R. Srikant, D. Thomas, Privacy preserving OLAP, SIGMOD (2005)
4. N. Alon, P. B. Gibbons, Y. Matias, M. Szegedy, Tracking join and self-join sizes in limited storage, PODS (1999)
5. N. Alon, Y. Matias, M. Szegedy, The space complexity of approximating the frequency moments, STOC (1996)
6. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, M. Y. Zhu, Tools for privacy preserving distributed data mining, SIGKDD Explorations (2002)
7. A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi, Processing complex aggregate queries over data streams, SIGMOD (2002)
8. W. Du, M. J. Atallah, Privacy-preserving cooperative statistical analysis, Computer Security Applications Conference (2001)
9. W. Du, Z. Zhan, Building decision tree classifier on private data, In Workshop on Privacy, Security, and Data Mining, ICDM (2002)
10. F. Emekci, D. Agrawal, A. E. Abbadi, A. Gulbeden, Privacy preserving query processing using third parties, ICDE (2006)
11. B. Goethals, S. Laur, H. Lipmaa, T. Mielikainen, On private scalar product computation for privacy-preserving data mining, International Conference in Information Security and Cryptology (2004)
12. O. Goldreich, Secure multi-party computation. Working draft, Version 1.3 (2001)
13. J. Gray, A. Bosworth, A. Layman, H. Pirahesh, Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals, ICDE (1996)
14. N. Jefferies, C. Mitchell, and M. Walker. A proposed architecture for trusted third party services. In Cryptography Policy and Algorithms Conference (1995)
15. M. Kantarcioglu, J. Vaidya, An architecture for privacy-preserving mining of client information, In Workshop on Privacy, Security and Data Mining, ICDM (2002)
16. National Institute of Standards and Technology (NIST), Secure hash standard, Federal Information Processing Standards Publication (FIPS) 180-2 (2002)
17. R. She, K. Wang, A. W. Fu, Y. Xu, Computing join aggregates over private tables. Technical report TR 2007-12, School of Computing Science, Simon Fraser University (http://www.cs.sfu.ca/research/publications/techreports/) (2007)
18. D. R. Stinson, Cryptography: theory and practice, Chapman & Hall/CRC, 3rd ed. (2006)
19. J. S. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. SIGKDD, 639-644 (2002)
20. L. Wang, S. Jajodia, D. Wijesekera, Securing OLAP data cubes against privacy breaches, IEEE Symposium on Security and Privacy (2004)
21. A. C. Yao, How to generate and exchange secrets. 27th IEEE Symposium FOCS (1986)
22. N. Zhang, W. Zhao, J. Chen, Cardinality-based inference control in OLAP systems: an information theoretical Approach, DOLAP (2004)